Problem 9 (20 Points)

Problem description

So far, we have worked with ~2 dimensional problems with 2-3 classes. Most often in ML, there are many more explanatory variables and classes than this. In this problem, you'll be training logistic regression models on a database of grayscale images of hand-drawn digits, using SciKit-Learn. Now there are 400 (20x20) input features and 10 classes (digits 0-9).

As usual, you can use any code from previous problems.

Summary of deliverables

- OvR model accuracy on training data
- OvR model accuracy on testing data
- Multinomial model accuracy on training data
- Multinomial model accuracy on testing data

Imports and Utility Functions:

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression

def visualize(xdata, index, title=""):
    image = xdata[index,:].reshape(20,20).T
    plt.figure()
    plt.imshow(image,cmap = "binary")
    plt.axis("off")
    plt.title(title)
    plt.show()
```

Load data

The following cell loads in training and testing data into the following variables:

- `x_train` : 4000x400 array of input features, used for training
- `y_train` : Array of ground-truth classes for each point in `x_train`
- `x_test` : 1000x400 array of input features, used for testing
- `y_test` : Array of ground-truth classes for each point in `x_test`

You can visualize a digit with the `visualize(x_data, index)` function.

In [9]:
```
x_train = np.load("data/w3-hw3-train_x.npy")
y_train = np.load("data/w3-hw3-train_y.npy")
x_test = np.load("data/w3-hw3-test_x.npy")
y_test = np.load("data/w3-hw3-test_y.npy")
```

In [9]:
```
x_train = np.load("data/w3-hw3-train_x.npy")
y_train = np.load("data/w3-hw3-train_y.npy")
x_test = np.load("data/w3-hw3-test_x.npy")
y_test = np.load("data/w3-hw3-test_y.npy")

visualize(x_train,1234)
```

Logistic Regression Models

Use sklearn's `LogisticRegression` to fit a multinomial logistic regression model on the training data. You may need to increase the `max_iter` argument for the model to converge.

Train 2 models: one using the One-vs-Rest method, and another that minimizes multinomial loss. You can do these by setting the `multi_class` argument to "ovr" and "multinomial", respectively.

More information: [https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

In [22]:
```
# YOUR CODE GOES HERE (sklearn models)
model_ovr_train = LogisticRegression(multi_class = 'ovr',max_iter = 50000)
model_ovr_train.fit(x_train,y_train)

model_multinomial_train = LogisticRegression(multi_class = 'multinomial',max_iter
model_multinomial_train.fit(x_train,y_train)
```

Out[22]:
```
LogisticRegression

LogisticRegression(max_iter=50000, multi_class='multinomial')
```

Accuracy

Compute and print the accuracy of each model on the training and testing sets as a percent.

Accuracy

Compute and print the accuracy of each model on the training and testing sets as a percent.

In [23]:
```
# YOUR CODE GOES HERE (print the 4 requested accuracy values)
preds = model_ovr_train.predict(x_train)
accuracy = np.sum(preds == y_train) / len(y_train) * 100
print("    Accuracy of train data for ovr:", accuracy, r"%")

preds = model_multinomial_train.predict(x_train)
accuracy = np.sum(preds == y_train) / len(y_train) * 100
print("    Accuracy of train data for multinomial:", accuracy, r"%")

preds = model_ovr_train.predict(x_test)
accuracy = np.sum(preds == y_test) / len(y_test) * 100
print("    Accuracy of test data for ovr:", accuracy, r"%")

preds = model_multinomial_train.predict(x_test)
accuracy = np.sum(preds == y_test) / len(y_test) * 100
print("    Accuracy of test data for multinomial:", accuracy, r"%")
```

```
    Accuracy of train data for ovr: 94.72500000000001 %
    Accuracy of train data for multinomial: 96.475 %
    Accuracy of test data for ovr: 90.7 %
    Accuracy of test data for multinomial: 91.4 %
```

In [ ]: